DEVELOPMENT OF A LINK LAYER PROTOCOL USING UML

K Chandra Sekaran,
Department of Computer Engineering,
Karnataka Regional Engineering College,
Surathkal-Srinivasnagar – P.O – 574 157, India.
Email: kch@krec.ernet.in

Abstract

This paper presents a report on the development of a link layer protocol using an object oriented formal method. It describes the steps involved in the conceptualization, analysis and design of the L2CAP (Logical Link Control and Adaptation Layer Protocol) of the Bluetooth architecture. UML (Unified Modeling Language) is the formal modeling language used in this work. It has been shown how the static, dynamic and operational aspects of the system are captured using various UML diagrams.

KEY WORDS: Bluetooth, protocol development, L2CAP, UML, formal method, object orientation.

1. Introduction

Some of the important developments in the recent past have been in the areas of mobile communication and object orientation. Formal method or, formal description technique (FDT) that is based on object orientation or object technology has proven to be a powerful tool for the design, development and testing of systems, particularly software systems, with strong correctness requirements [1-3]. The basic idea of object orientation is to capture objects which model the real-world processes accurately.

The Unified Modeling Language (UML) [4,5] is a visual, highly expressive language for describing the objects and their relationships of a system. Even though there were attempts in providing a feel of object orientation [6] in FDTs such as Estelle, due to the additional effort of learning and using these techniques, they have not been widely adopted in the industry environment. Also, most of the popular FDTs were used on specification of protocols without giving any importance to how these specifications will lead to the development of the protocol in terms of a product as a whole [7]. Hence, the goal of the research that is described here is how to use UML as the formal description technique and present the experience of developing a link layer protocol, specifically L2CAP layer of Bluetooth [8] architecture.

This paper is organized in the following manner. Section 2 gives a brief introduction of UML and its expressive power in terms of its diagrams. Section 3 describes the Bluetooth architecture in short and views L2CAP's role as a protocol in mobile communication. Section 4 provides the object oriented analysis and design of the above said protocol using UML and outlines the implementation notes, and section 5 provides the conclusion.

2. Unified Modeling Language

UML is more complete [9] than other formal methods in its support for modeling any complex system including communication systems. The system's requirements and design are represented a high level using class models, use-case models, and state/activity diagrams. Its major features include: Object Model, Use Cases and Scenarios, Behavioral modeling with statecharts, Packaging of various kinds of entities, Representation of tasks and their synchronization, Models of physical topology, Models of source code organization, and Support for object oriented patterns. The following paragraphs outline the a few important features of UML. More details of UML can be obtained from [5].

Objects form the fundamental unit of object-oriented systems. An object is, roughly speaking, any type of real life entity. Objects bind together data (represented as attributes) with operations that act on that data. The UML diagrams that correspond to objects identify the objects themselves and their arrangements into collaborations (via association relations among them) and into taxonomic hierarchies (via generalization relations among them). Use-case diagrams represent the interaction of a system and/or its components with the external environment or other subsystems. The objective is to identify the external interfaces clearly. There is no direct, obvious mapping of the use-case model to the object model. In general, a use-case is realized by a collaboration of objects working together. This is complicated by the fact that an object can participate in multiple collaborations. Typical examples of some of the diagrams of UML are given here (see Figures: 4 – 7). As it is shown, the ovals are the use-cases-chunks of functionality visible to one or more actors that do not reveal or imply the internal

0-7695-1381-6/01 $10.00 © 2001 IEEE

structure necessary to implement that functionality. Actors are objects that are outside the scope of the system but that interact with the system. Behavioral models, particularly sequence diagrams, are used to elaborate the details of the system-actor interaction.

A statechart is a popular way of providing a visual formalism of behavioral model. A state is a condition of existence of an object (or, more generally, of a UML metaclass, called a "classifier") that persists for a significant period of time. An object's transitions among states in response to received events (occurrences in time and space that are of interest to the object). Transitions are shown by directed arcs connecting the states. Statecharts also include the ability to nest states (deal with states at multiple levels of abstraction) and to divide the behavior of an object up into independent regions called "and-states" (and-states are separated by dashed lines). Behavior of collaborations may be defined by statecharts associated with the use-case or, more commonly, scenarios can be shown by sequence diagrams.

3. Bluetooth

3.1 Introduction to Bluetooth

The Bluetooth wireless technology is a standard for wireless communication between computers, printers, telephones and other electronic devices over short distances. It aims at allowing users to make fast connections between communication devices, such as mobile phones and computers. It allows high quality voice and data transmissions. The main features are less complexity, low power and cost. Bluetooth radio modules avoid interference from other signals by hopping to a new frequency after transmitting or receiving a packet. Compared to other systems operating in the same frequency band, Bluetooth radio hops faster and uses shorter packets. This makes Bluetooth radio robust as compared to other systems. The Bluetooth special interest group includes companies namely 3Com, Ericsson, IBM, Intel, Lucent, Microsoft, Nokia and Toshiba.The Bluetooth wireless technology provides a transmission speed of 1 Mbit/s and handles up to three voice channels simultaneously. It operates at 2.4 GHz. The data is transmitted at the rate of 1Ms/s. It uses binary frequency shift keying modulation. The frequency deviation is between 140 to 175 kHz. It supports both point-to-point and point to multipoint connections. A piconet has up to eight connected devices. One device is called the master and rest slaves. Up to seven 'slave' devices can be set to communicate with a 'master' .The master's clock and hopping sequence are used to synchronize all devices in the piconet. Many of these 'piconets' can be linked together to form 'scatternets'. All devices in the same piconet have priority synchronization, but other devices can be set to enter at any time.

3.2 The block diagram of Bluetooth protocol stack

The Bluetooth architecture or protocol stack is shown below – figure-1. As it can be seen from this figure, the bluetooth has four layers: application layer, network layer, link layer consisting of the following components – LMP (Link Manager Protocol), L2CAP (Logical Link Control and Adaptation Layer) and HCI (Host Controller Interface), and the physical layer (Baseband). Components of link layer are briefly explained in the following paragraphs.

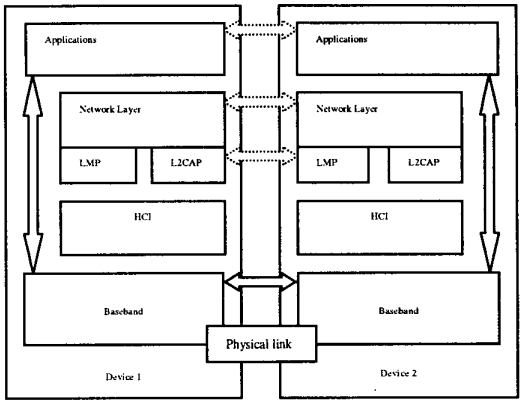

Figure 1: Bluetooth protocol stack

The Baseband protocol is a combination of circuit and packet switching. Slots are reserved for synchronous packets. Each packet is transmitted in a different hop frequency. A packet usually covers a single hop but can extend upto five slots. Baseband allows both synchronous connection oriented (SCO) and asynchronous connectionless links (ACL). SCO is for voice traffic whereas ACL is for data.

The HCI provides a command interface to the Baseband controller and link manager and access to hardware status and control registers. The LMP messages are used to set up a link, security and control. The messages are filtered out and interpreted by LM on the receiving side and are not propagated to higher layers. Link manager messages have higher priority than user data. This means that if the link manager has to send a message, it is not delayed by the L2CAP traffic.

The logical link control and adaptation layer protocol – L2CAP is above the baseband protocol and resides in the data link layer. L2CAP provides

both connectionless and connection oriented data services to the upper layers along with segmentation and reassembly operation, protocol multiplexing capability and group abstractions. It allows higher level protocols to transmit and receive L2CAP data packets upto 64 Kilobytes in length.

3.3 L2CAP in bluetooth protocol architecture:

The following Figure - 2 depicts the way in which L2CAP fits in the protocol stack of bluetooth:

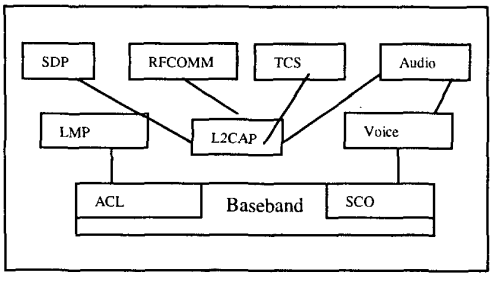

Figure 2: LECAP in Bluetooth

L2CAP helps in transfer data between higher layer protocols (Service Discovery Protocol, RFCOMM, and Telephony Control Service) and lower layer protocol (Host Controller Interface). For each implementation there is a support for set of signalling commands. Events should be accepted by L2CAP and in turn generate events for the upper layer. Passing of these events is depends on the implementation. Segmentation and Reassembly (SAR) operations are used to improve efficiency by supporting maximum transmission unit (MTU) size larger than the Baseband packet.

Large L2CAP packets may be segmented into smaller packets before they are transmitted over the air. L2CAP also does the protocol service multiplexing because the Baseband does not have any type field to identify the higher layer protocols multiplexed over it.

SDP – Service Discovery Protocol is used by the applications to discover which services are available and to find the characteristics of those available services. It allows clients to search for the needed services based on specific attributes of those services. It also allows services to be discovered based on class of service. Further, it allows to discover new services, which become available when devices enter RF proximity with a client device. It also allows a client on one device to discover a service on another device without consulting the third device. The function of telephony control service (TCS) is to provide call control. This means, there is signaling for (a) the establishment and release of speech and data calls between two Bluetooth devices, (b) group management, to handle groups of Bluetooth devices, and, (c) connectionless TCS which is used to exchange signaling information without setting a TCS.

3.4 Functions of L2CAP

The L2CAP essentially provides 2 types of functions: (a) signaling between L2CAP entities of different mobile devices, and, (b) data transfer operation between layers – upper and lower layers. The signaling between the entities of logical link control and adaptation protocol (L2CAP) on mobile devices is based on the concept of channels. A channel identifier knows each one of the end points of an L2CAP channel. Channel identifiers (CIDs) are local names representing a logical channel end point on the device. The null identifier 0x0000 is defined as an illegal identifier and must not be used as a destination end point. CID assignment is relative to a particular device and a device can assign CIDs independently from other devices. In this signaling, there are three types of communications possible. That is, the ways in which the CIDs can be used to communicate between L2CAP entities in separate devices could be of any one of the following: (a) the connection oriented data channel, (b) the connectionless data channel which restricts the data flow to a single direction, and, (c) the signaling channel, which is an example of reserved channel and is used to create and establish connection oriented data channels and to negotiate if there is any change in the characteristic of these channels. Thus, signaling and its commands must be used between the L2CAP entities.

The L2CAP implementation must also require to do transfer the data between higher layer protocols and lower layer protocol. L2CAP should accept some events from lower layers and generate events to the upper layer. Segmentation and reassembly in L2CAP allows the transmission beyond the largest baseband packet between layers. L2CAP divides the packet into chunks that can be passed to the link manager via the host controller interface. On the receiving side, the L2CAP reassembles those chunks into L2CAP packets again using the information given in HCI and the packet header. These functions are depicted as layer interactions in the following Figure – 3.

The client and server shown in the Figure 3 represent the initiator and acceptor of the request at different devices. At the application level client will initiate and server accepts the requests.

Events of upper layer are called Requests and the replies corresponding to them are Confirms. Events of lower layer are called Indications and the replies are called Responses. Responses which have to be processed further are called Pending. Confirms and Responses indicate positive replies. A request for an action results in a Confirmation and Indications need not result in responses. The naming convention that is used in Figure – 3 for the requests and responses, is to identify distinguish the message type between layers, particularly lower to upper.

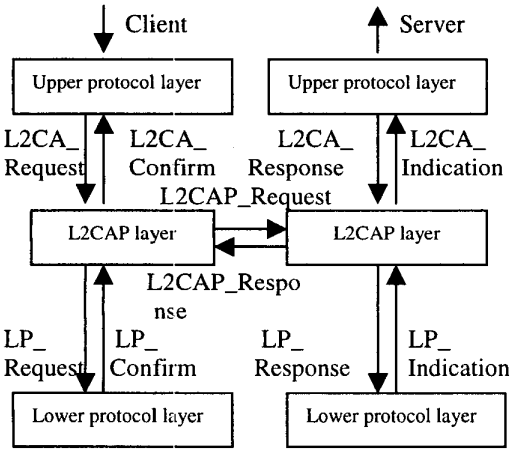

Figure 3: L2CAP Layer Interactions

4. Developing L2CAP using UML

This work is based on the classical methodology for software development using formal description techniques. In this research, UML is used as a formal method. First the informal specifications of the requirements of the functions or services of L2CAP, describing the interfaces with the upper and lower level layers, and, the signaling between the L2CAP entities among the devices. The next step is developing the formal specification of the protocol in UML based on the informal specification. The formal specification essentially consisted of (a) object models which abstract the functions of the L2CAP, (b) the methods that correspond to the processes or actions in describing the functions of L2CAP, and, (c) the static and dynamic behavior of L2CAP as actions in responding to various events. The following paragraphs describe these steps while applying UML.

4.1 Software Development Process

The Rational Rose Process (RUP) [10] is the recommended software process for this object oriented protocol development.

The following workflows have been used in the process.

- Requirement Analysis is carried out to bring out all important use cases, their descriptions with the use-case diagrams.
- Domain Analysis is performed to discover all important abstractions or object models in this L2CAP protocol design and development problem domain, which includes the classes, interfaces, relationships and collaborations . Also this step involves the development of all relevant structural and behavioral models and their UML diagrams.
- Design involves the architectural design and detailed design of the software that implements the protocol. The following issues are being considered here: the user interface to enable the network administrator to monitor the system and the database component to keep track of some of the house keeping activities such as the number of connections and their details.
- Implementation and Testing involves the simulation of upper layers to test the functionality of the implemented L2CAP with its specifications.

4.1 Requirements Analysis

The textual, informal requirements specification for L2CAP is as follows:

- Signaling between L2CAP entities of different mobile devices: This involves
- Identifying channels on mobile devices using CIDs
- Signaling between devices using CIDs in any one of the communication modes:

a) the connection oriented data channel
b) the connectionless data channel which restricts the data flow to a single direction, and
c) the signaling channel which is used to create and establish connection oriented data channels and to negotiate if there is any change in the characteristics of these channels.

- Data transfer operation between layers: This involves
- Handling requests of upper/lower layers, and,
- Sending responses to upper/lower layers.

While handling this request/response data of upper and lower layers, it becomes necessary to do the following too:

- Segmentation and Re-assembly of protocol data units (PDU).

4.2.2 Use Case Analysis

The L2CAP protocol's functionality are captured through the use cases. Firstly we identify the actors of the system as: Upper Layer, Lower Layer, Peer Entity and Network Administrator. Secondly, the use cases for each of these actors have been identified by considering the functions that are required for actors from the system. The use cases thus identified are: Request, Reply, Segmentation, Reassembly, Signaling and Admin.

Figure 4 shows a simplified version of UML use case diagram depicting the relationships among these use cases. As this diagram is self-explanatory, further descriptions are not provided. Also detailed use case diagrams for different use cases of this figure are too not provided.

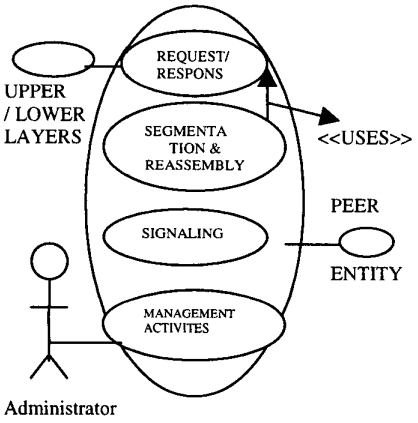

Figure-4 Use Case Diagram for LECAP

4.3 Domain Analysis

4.3.1 Class Diagram

The information gathered during the construction of the use cases has been used to perform object decomposition and build the object structure or so called class diagram of the system.

- Object Identification: Major objects identified are: Layers, Admin, Message Type, Connection Mode and PDU Operation. The next level of key objects are: Upper Layer, Lower Layer, Peer Entity, Connectionless Mode, Connection Oriented Mode, Connection Signal, Segmentation, Reassembly, Request, Response, Confirm and Indication.
- Relationships: The relationship among the major classes and key classes are as follows: Upper Layer, Lower Layer and Peer Entity are kind of Layers. Connectionless Mode, Connection Oriented Mode and Connection Signal are the kinds of Connection Mode. Segmentation and Reassembly are kind of PDU Operation. Request, Response, Confirm and Indication are kind of Message Type. Thus, all these fall into the category of ***is a*** relationship. Individual Message Type, Connection Mode, Layer and Admin are part of the protocol and hence they fall into the category of ***aggregation*** relationship.
- Object attributes: The following strategies have been used to discover the attributes of the identified objects: (a) information to define the object (b) information with regard to the methods that correspond to the object and (c) information that are necessary to fulfill the responsibilities of the object.

The above steps lead to the development of complete UML class diagram for L2CAP design. However, as the presentation is restricted to limited number of pages and considering the fact that drawing each section of the class diagram separately will occupy too many pages, only a section of the class diagram is shown in Figure – 5.

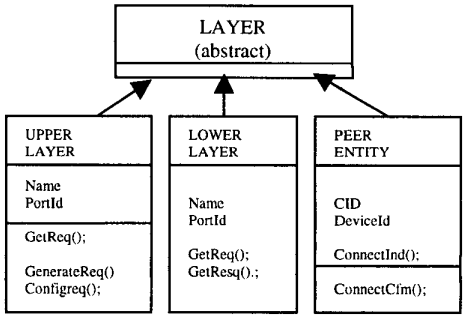

Figure – 5 UML Class Diagram – partial view

4.3.2 Other UML Diagrams

The domain analysis continues with the development of other UML diagrams – viz., Sequence diagrams, State Charts and State transition diagrams, Activity diagrams, Collaboration diagrams, Component diagrams and Deployment diagrams. Of these, Component diagrams and Deployment diagrams are used at the implementation phase. In this presented research work all these diagrams have been developed.

4.3.2.1 Sequence Diagrams

Sequence diagrams describe interactions among classes. These interactions are models as exchange of messages. These diagrams focus on classes and the messages they exchange to accomplish some assigned behavior. In our L2CAP. Figure 6 depicts a simplified view of a Sequence diagram used in the design of L2CAP.

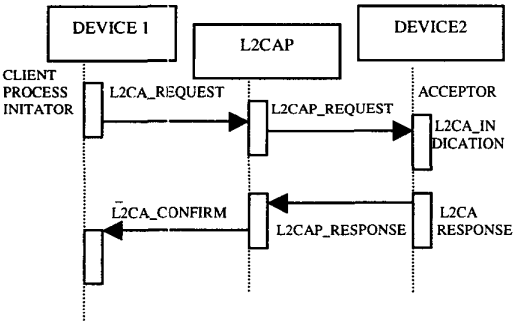

Figure-6 Sequence diagram for communication between devices

4.3.2.2 State Transition Diagrams

State transition diagrams describe the states and responses of a class. State-chart diagrams describe the behavior of a class in response to external stimuli. Typical operational states and the corresponding state transition diagram is shown in Figure 7. In the design of L2CAP, we have three major states, namely Closed, Open and Config, and, there are four minor states. The terms major and minor indicate the important and intermediate states. Figure 7 shows only major states. In the Closed State, there is no channel associated to a CID. This is the state where in a link level connection may not exist. In other words, link disconnection puts all other states into this state. In the Config State, the connection has been established but both sides are still negotiating the channel parameters. The Configuration state may also be entered when the channel parameters are being renegotiated. Before entering the Config state, all outgoing data traffic should be suspended as the traffic parameters of the data traffic are to be renegotiated. Incoming data traffic must be accepted until the remote channel endpoint has entered the Config state. From the Config state to the OPEN state requires both sides to be ready. An L2CAP entity is ready when it has received a positive response to its request and it has positively responded to the final request from the remote device. In the Open State, the connection has been established and configured, and data flow may proceed.

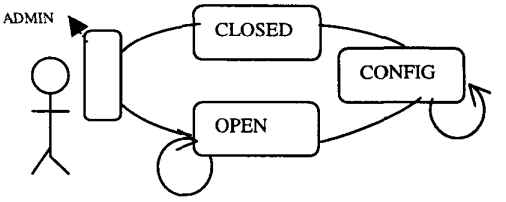

Figure 7 State Transition Diagram

4.4 Object Oriented Design

Object oriented design is the process of specifying an implementation that is consistent with the analysis model. This phase enabled to give a high level design of L2CAP, where subsystems (packages) are defined, including dependencies and communication mechanisms between the packages. In our context, the following points have been takn into consideration as the technical implementation issues in the development of the architecture:

- Introduction of the concept of Local Device
- This Local Device module of the architecture handles all the issues related to the different layers in the local computing device in which the L2CAP software is housed. They include:
  - Multiplexing between applications of upper layers
  - Handling request messages of upper layers
  - Handling response messages of lower layers
  - Segmentation and reassembly of PDUs of upper and lower layers
- Introduction of the concept of External Device
- This External Device module of the architecture handles all the issues related to the different external devices that have connectivity with the local computing device in which the L2CAP software is housed. They include:
  - Management of different connection details with CIDs
  - Authentication

Thus, a two tier architecture to support the above said modules has been proposed and followed. Following are the packages in this proposed architecture:

(a) Up_Layer – consists of files corresponding to the upper layer interaction with L2CAP.
(b) Lo_Layer – consists of files that correspond to the activities of lower layer.
(c) Peer_Layer – consisting the files of external device connectivity

(d) Admn – enable the network administrator to probe for the debugging activities.

4.5 Implementation

L2CAP software has been implemented as a two tier architecture as described in section 4.4, above, with appropriate component and deployment diagrams. Java language has been used for this purpose. The simulated version of this software uses JDBC and MS-ACCESS for storing and retrieving data. Some of the efficiency tips that have used in implementing the functionalities of L2CAP are as follows:

- Instead of using two buffers, one between the upper layer and L2CAP and other between the L2CAP and lower layer, using only a single buffer.
- To distinguish between the requests and responses, while checking for the first field that is the code field, check whether it is even or odd. All even ones are requests and odd ones responses.
- While inserting a node in the connection list, insert it in the ascending order so that the channel identifiers are allocated from the free pool and traversing becomes easier and saves time while searching for a particular node in the entire list.

4.5.1 Testing

After the development of the protocol stack for L2CAP, all the functions need to be tested individually and then they have to be tested after integration. Since the L2CAP layer is the middle layer in Bluetooth, it can be tested efficiently with the presence of the upper and lower layers, and hence the simulation of these layers has been done for testing purpose.

5. CONCLUSION

The project aims at developing the protocol stack for Logical Link Control and Adaptation Protocol. The functions pertaining to the protocol stack of L2CAP have been implemented using object oriented analysis and design. The upper and lower layers were simulated for testing purpose.

The contribution of the work includes finding the input and output parameters for L2CAP, host controller interface, developing the code for L2CAP in an object oriented manner designed using UML, upper and lower layer simulation. The upper layers being service discovery protocol, RFCOMM and Telephony Control Service and, the lower layer includes winsock programming.

Due to paucity of space, this paper could not cover all details of object oriented analysis and design that is carried out using UML in the development of L2CAP. Interested readers may contact the author by email for more details.

6. REFERENCES

[1] G. Booch. Object Oriented Analysis and Design with Applications. Second Edition. Benjamin Cummings, 1994

[2] G. Booch, J.Rumbaugh, and I.Jacobson. The Unified Modeling Language User guide. Addison Wesley Longman, 1999.

[3] Jonathan Weinstock , and Rajiv Tewari, An Object Oriented approach to the management of distributed application systems, Computer Networks and ISDN Systems 29 (1997) 1869 – 1879.

[4] H.E.Eriksson and M.Penker. UML Toolkit. Weiley Computer Publishing, 1998.

[5] www.rational.com website

[6] B.Prabhakaran and S.V.Raghavan, Object oriented extensions to ESTELLE, proc. of Intl.conf.on Computer Communications, ICCC 90, pp. 750 – 758.

[7] Anand R.Tripathi, and Terence Noonan, Design of a RPC for Object Oriented Distributed Programming, Software – Practice and Experience, vol. 28(1), Jan.1998, pp.23-47.

[8] www.bluetooth.com website

[9] Bruce Powel Douglass. Real-Time UML: Developing Efficient Objects for Embedded Systems, Addison Wesley Longman, 1999.

[10] I.Jacobson, G.Booch and J.rumbaugh. The Unified software Development Process. Addison Wesley Longman, 1999.